RLE-BENCH: A QUALIFYING EXAM FOR CODING AGENTS AS ROBOT LEARNING ENGINEERS

Anonymous authors
Paper under double-blind review

ABSTRACT

Coding agents are beginning to move beyond purely digital tasks to tackle physical-world challenges, particularly in robotics. Existing robotics benchmarks, however, primarily focus on the performance of individual artifacts, such as policies or controllers, offering limited coverage of coding agents' broader engineering capabilities. Real-world robotics extends beyond control: agents must build, integrate, diagnose, and improve heterogeneous artifacts under resource constraints and reason from multimodal feedback. To evaluate these broader capabilities, we introduce RLE-BENCH, a benchmark of robot-learning tasks spanning four representative robotics development workflows: interactive control, policy learning, perception and estimation, and mechanical design. We use diverse task-specific metrics to evaluate the artifacts submitted by the coding agents, from the success rate the agents achieved to the policy agents trained, the harness agent built, and the mechanical structures the agent designed. We aggregate these metrics into an overall RLE Index and report workflow-specific capability profiles, enabling systematic comparison of coding agents' capabilities across multiple capability dimensions. Beyond performance ranks, we also conduct in-depth case studies examining agent behavior on representative tasks, highlighting both current capabilities and limitations, and pointing to the opportunities robotics tasks have to offer for future agent training.

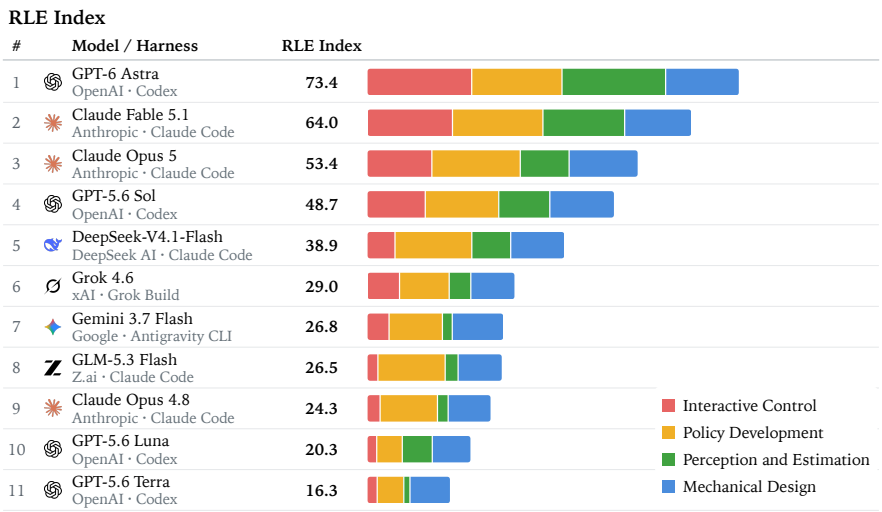


Figure 1: **RLE-BENCH Leaderboard.** Models are ranked by the overall RLE Index.

1 INTRODUCTION

Coding agents have moved well beyond code completion. They can now inspect unfamiliar repositories, write and execute programs, interpret test failures, and revise artifacts over long trajectories.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

Benchmarks have played a key role in this transition, both by defining the need and by providing signals for how to scale up training. For example, repository-level coding benchmarks such as SWE-bench (Jimenez et al., 2024) and interactive environments such as Terminal-Bench (Merrill et al., 2026) have made this transition measurable, while machine-learning engineering benchmarks (Huang et al., 2023a; Chan et al., 2025; Qiang et al., 2025) further test their capability in training, evaluating, and improving machine learning systems at a larger scale.

Beyond digital engineering tasks, interacting with and reasoning about the physical world remains the grand challenge for LLM agents. Unlike purely digital settings, robotics requires agents to understand spatiotemporal relationships, reason about the physical consequences of their decisions, and adapt their actions based on multimodal feedback from the environment. Robotics engineering, which covers diverse topics such as mechanical design, hardware-software calibration, closed-loop control, failure diagnosis, and policy development, provides an excellent testbed to showcase these physically grounded capabilities. However, existing robotics benchmarks (Liu et al., 2023; Mu et al., 2025; Chen et al., 2026) primarily evaluate policies manifested as machine learning models, such as vision-language-action models (VLAs, Brohan et al. (2022); Black et al. (2024)) and world-action models (WAMs, Ye et al. (2026)), rather than the broader capabilities coding agents exercise throughout the robotics development process. As an early step toward evaluating coding agents in robotics, CaP-X (Fu et al., 2026) systematically benchmarks an agent's ability to write code as controllers for manipulation tasks across different levels of abstraction; however, its evaluation remains limited to writing code as policies rather than the holistic robot-learning workflow.

To fill this gap and measure whether frontier agents can transfer their capabilities into real-world physical tasks, we introduce RLE-BENCH, which asks:

> *Can general-purpose coding agents qualify for the diverse, iterative, and physically grounded work of robot-learning engineers?*

The benchmark comprises nine families of tasks organized into four primary workflow families: *Interactive Control*, *Policy Learning*, *Perception and Estimation*, and *Mechanical Design*. Each task provides an instruction and an interactive simulator through which the agent can observe, act, and receive physical feedback. The resulting agentic systems or executable artifacts are evaluated using benchmark-owned instrumentation under hidden scenes, dynamics, embodiments, random seeds, or held-out tasks.

A key distinction of RLE-BENCH from existing robotics benchmarks is its emphasis on a broader range of robot-learning workflows: rather than evaluating policies, controllers, or other components in isolation, RLE-BENCH evaluates whether coding agents can perform and improve the heterogeneous tasks that collectively constitute robotics development. This holistic perspective is important for the future we envision, in which coding agents can organically integrate capabilities across the development stack and use them to iteratively improve both their robotic systems and the infrastructure used to build them.

We evaluate 11 model–harness combinations and report the **RLE Index**, an equally weighted macro-average of the four workflow-family scores in the benchmark, as our primary metric for each combination. Beyond this scalar metric, we provide per-workflow profiles that illustrate the performance and cost trade-offs across the capabilities we care about for robotics. We also conduct in-depth case studies examining agent behavior on representative tasks, highlighting both current capabilities and limitations, and pointing to the opportunities robotics tasks offer for future agent training.

2 RELATED WORK

Software Engineering (SWE) and Machine Learning Engineering (MLE) Benchmarks. To faithfully evaluate agents' coding capabilities and provide interactive testbeds in which they can act and learn from feedback, a growing number of coding-oriented benchmarks have emerged. Among them, SWE-Bench (Jimenez et al., 2024) curates real-world repositories and issues from the web and tasks agents with generating code patches, which are evaluated against functional tests to provide feedback on their actions. OSWorld evaluates agents in interactive computer environments using task-

Table 1: Comparison of benchmark evaluation scope. The SWE / MLE group includes SWE-bench, Terminal-Bench, MLE-bench, and MLE-Dojo.

Benchmark	Coding Agents	Physics Grounded	Controller Synthesis	Robot Policy Training	Perception and Estimation	Mechanical Design
SWE/MLE benchmarks	✓	✗	✗	✗	✗	✗
LIBERO/RoboTwin/RoboDojo	✗	✓	✗	✓	✗	✗
CaP-X	✓	✓	✓	✗	✗	✗
RLE-Bench	✓	✓	✓	✓	✓	✓

specific execution-based checks (Xie et al., 2024a), while Terminal-Bench evaluates long-horizon terminal tasks in isolated execution environments (Merrill et al., 2026). Beyond traditional SWE tasks, recent work has increasingly focused on machine learning engineering. MLAgentBench and MLE-Bench shift the focus from software repair to experimental machine learning, requiring agents to train, evaluate, and improve machine learning models (Huang et al., 2023a; Chan et al., 2025). MLE-Dojo provides a Gym-style environment spanning over 200 Kaggle challenges, with structured interaction, online outcome verification, and support for both agent evaluation and training (Qiang et al., 2025). RLE-BENCH builds on this paradigm of executable, iterative evaluation while extending it to robot learning, where agent actions are grounded in physical consequences and directly interact with physical simulators.

Robotics Policy Evaluation Benchmarks. Numerous benchmarks have been developed to facilitate the prototyping and evaluation of robot policies. These benchmarks typically provide physics simulators, task environments, and evaluation protocols, and can be broadly categorized by task domain. RLBench (James et al., 2020), CALVIN (Mees et al., 2022), LIBERO (Liu et al., 2023), RoboSuite (Zhu et al., 2020), RoboCasa (Nasiriany et al., 2024), RoboTwin (Mu et al., 2025; Chen et al., 2025), and RoboDojo (Chen et al., 2026) focus primarily on manipulation with fixed-base or mobile manipulators. Gym-Locomotion (Brockman et al., 2016) and DMControl (Tassa et al., 2018) cover locomotion, while broader platforms such as IsaacLab (Mittal et al., 2025) and ManiSkill (Tao et al., 2024) support manipulation, locomotion, and whole-body control. However, these benchmarks primarily evaluate individual artifacts, such as controller or VLA policies, rather than the broader capabilities of coding agents in performing robotics development workflows.

Coding Agents for Robotics. A predominant use of coding agents in robotics is to generate controller code for robot control. Code as Policies (Liang et al., 2023) and VoxPoser (Huang et al., 2023b) demonstrate that language-model-generated programs can solve manipulation tasks. Building on this paradigm, CaP-X systematically evaluates code-as-policy agents across varying levels of abstraction, interaction, and perceptual grounding (Fu et al., 2026). Concurrent work explores coding agents for improving policy repositories, reproducing robot-learning systems, and operating physical experimentation loops (Elmaaroufi et al., 2026; Xiao et al., 2026; Jin et al., 2026). Beyond controller generation, LLM agents have also been used to generate reward functions for reinforcement learning, as exemplified by Text2Reward (Xie et al., 2024b) and Eureka (Ma et al., 2024). In contrast to these specialized applications, RLE-BENCH takes a holistic perspective, systematically evaluating coding agents across the broader robot-development stack.

Table 1 compares RLE-BENCH against existing benchmarks along each dimension.

3 RLE-BENCH

In this section, we detail the design of RLE-BENCH, covering its task templates, budget control, design principles of each workflow, and the evaluation protocol. Figure 2 provides an overview of the tasks in RLE-BENCH.

3.1 TASK DESCRIPTION AND EVALUATION SETUP

A benchmark task in RLE-BENCH is a tuple

$$\tau = (I, E_{\text{dev}}, B, \mathcal{A}, E_{\text{eval}}, M), \tag{1}$$

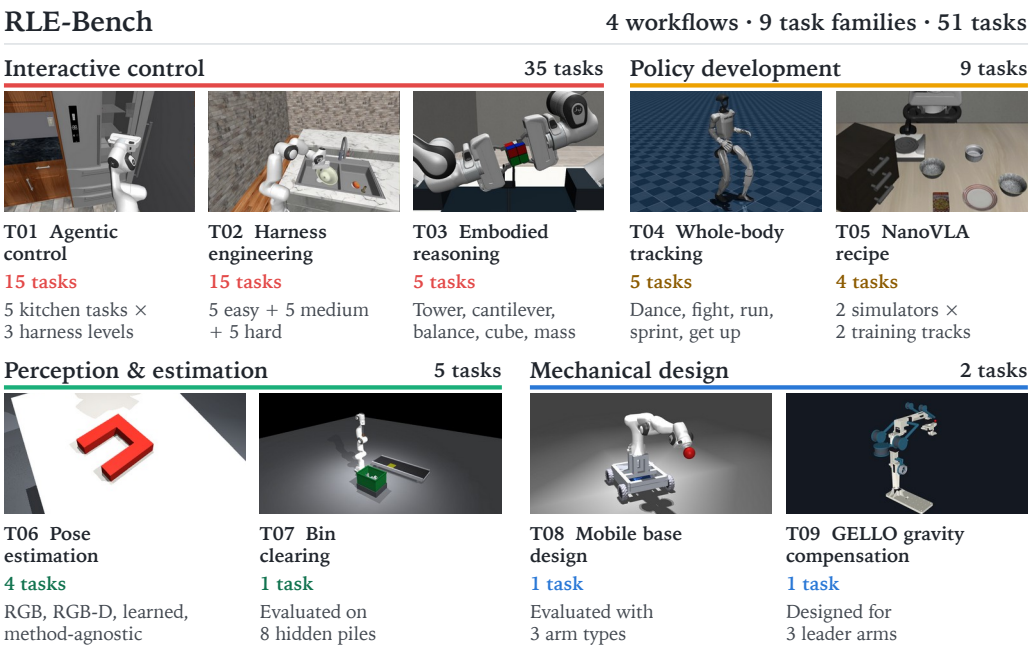


Figure 2: Overview of the RLE-BENCH.

Table 2: The RLE-BENCH task categories. A total of 51 tasks are categorized into 4 workflows, 9 families. Each family shares a similar capability requirement, artifact contract, and the evaluation metrics.

Task Family	Workflow	Objective	Submitted artifact	Metrics
`Family01`	Interactive control	RoboCasa task learning	Agent context	Success rate
`Family02`	Interactive control	Reusable RoboCasa harness	Harness code + manual	Success rate
`Family03`	Interactive control	Tabletop physical reasoning	–	Solution optimality
`Family04`	Policy learning	Whole-body motion tracking	Model checkpoint	Motion tracking error
`Family05`	Policy learning	VLA recipe engineering	Model checkpoint	Validation loss
`Family06`	Perception & estimation	Blind multi-shape pose estimation	Pose estimator code / models	Estimation error
`Family07`	Perception & estimation	Contact-rich bin clearing	End-to-end controller code	Operation efficiency
`Family08`	Mechanical design	Common mobile-manipulator base design	MJCF design + controller code	Design metric
`Family09`	Mechanical design	GELLO gravity compensation module design	MJCF design + controller code	Design metric

where I is the task instruction in plain text, E_{dev} is the public development environment such as a scratch workspace with connections to a RoboCasa simulator (Nasiriany et al., 2024), B is the budget including total wall clock time, interaction steps, compute resources, and etc. $\mathcal{A}$ is an executable artifact contract, E_{eval} is a verifier-controlled evaluation environment, and M is a verifier to evaluate the artifact submitted by the agent.

RLE-BENCH has 51 tasks in total. We organize them into four primary workflows according to the focused capabilities: *Interactive Control* (`Family01` $-$ `03`), *Policy Development* (`Family04` $-$ `05`), *Perception and Estimation* (`Family06` $-$ `07`), and *Mechanical Design* (`Family08` $-$ `09`). They are further categorized into 9 families according to the actual task objective, submitted artifacts, and evaluation metrics, which is summarized in Table 2. A detailed walkthrough of each task is provided in Section 3.2.

Each task in the suite is split into a development phase and an evaluation phase. In the development phase, the agent interacts with environment E_{dev} under resource constraints B and builds and submits an artifact a. In the evaluation phase, the verifier M evaluates the submitted artifact a in the evaluation environment E_{eval}. To receive a valid score, the artifact is generally required to satisfy the contract specified by the instruction I.

We construct RLE-BENCH from two complementary sources. The first category, comprising `Family01`, `02`, `04`, `05`, draws on simulators, datasets, and task definitions from existing open-source

benchmarks and projects, which we repurpose under new objectives to enable a standardized, comprehensive evaluation of coding agents. The second category, comprising `Family03`, `06`, `07`, `08`, `09`, is hand-designed by us, drawing inspiration from daily robotics engineering practice.

3.2 A Walkthrough of the Workflows and Task Families

In this section, we introduce the details about each task and its design motivations.

Workflow 1: Interactive Control (`Family01 − 03`). Before an agent can build anything for a robot, it has to be able to operate one. Direct interaction is exactly where physical grounding is most exposed, since observations are partial, actions cannot be undone, and progress depends on reading multimodal feedback correctly. In these tasks the agent interacts with the world directly to achieve given goals, either by controlling the robot itself, by building the tools that let another agent do so, or by acting to gather the evidence a decision needs.

Specifically, `Family01` evaluates agents to perform closed-loop control on 5 RoboCasa (Nasiriany et al., 2024) tasks under three different levels of harness that progressively add reusable utilities and privileged simulator state. `Family02` instead asks agents to package their experience into reusable tools and lessons, which will be used by fresh agents to solve held-out unseen tasks. `Family03` focuses on embodied reasoning, where agents must interact with the environment and gather information to infer task-critical physical properties, rather than relying on static visual QA.

Workflow 2: Policy Learning (`Family04 − 05`). Direct control does not scale to complicated tasks such as dexterous manipulations and high-frequency locomotion. In such scenarios, we must resort to a learned policy or controller, and the training recipes, such as the dataset, reward function, curricula, and model architecture, are the key to the final policy performance. Besides, policy learning is also another way for the agent to turn solved instances and experiences into durable, reusable capability.

`Family04` tackles locomotion, asking the agent to train humanoid whole-body controllers through motion tracking. The agent is provided with motion clips from the LAFAN1 dataset (Harvey et al., 2020) and must set up the entire training pipeline to train the tracking policy, which is then evaluated Sim2Sim under hidden variations in dynamics, latency, and sensor noise. `Family05` is named *nanoVLA*, inspired by the nanoGPT project for rapid iteration on training smaller-scale language models. We select LIBERO (Liu et al., 2023) and RoboTwin (Chen et al., 2025) and their paired datasets, ask the agent to develop the training pipeline and VLA architecture under resource constraints, and evaluate the delivered model on a selected subset of benchmark tasks.

Workflow 3: Perception & Estimation (`Family06 − 07`). Sensors are central to robotic systems, and coding agents must understand them well to make good decisions. Real perception systems are noisy, frequently occluded, and rate-limited, so these tasks directly evaluate how well agents understand the physical world through sensor readings.

`Family06` consists of four 2D pose estimation problems in a tabletop block-pushing task, where the block can move and become occluded by the robot arm, creating significant difficulty for pose estimation and requiring agents to reason about when occlusion occurs and how to handle it; to further constrain the scope, we define four variants in which agents are supplied with different sensors and inference hardware configurations. `Family07` asks agents to write a code-as-policy controller for a simplified industrial pick-and-place task, where a manipulator equipped with two RGB-D cameras, force-torque sensors, and a magnetic gripper must pick *all* U-shaped metal components from a bin onto a conveyor belt as fast as possible.

Workflow 4: Mechanical Design (`Family08 − 09`). Hardware bounds what any controller can achieve. An under-powered base or an uncompensated arm cannot be fixed in software, so an engineer has to reason about mass, torque, geometry, and build-ability together with the control that will run on them. This workflow evaluates mechanical design and its supporting control software as a coupled system.

`Family08` asks agents to develop a common mobile-manipulator base and controller for multiple robot arms including Franka Research 3, XArm7 and UR5e. The agents need to achieve the goal of reaching all the reference points on a shelf, while balancing payload, resource use, and stability.

`Family09` requires adding gravity compensation to GELLO lead-arms (Wu et al., 2024), evaluated under unseen physical instances, poses, and payloads. Both tasks make mechanical design and its supporting software part of the solution, rather than treating the embodiment as fixed.

3.3 Orchestration and Evaluation Protocol

To support standardized evaluation and integration, all tasks in RLE-Bench are shipped as Harbor environments (Team, 2026). Each agent receives the task instruction, a declared artifact contract, public assets, tools, and a resource-bounded development session. Once development ends, the artifact crosses into the verifier image, where the hidden seeds, privileged states, and reference assets all remain verifier-private to prevent information leakage. The verifier loads the artifact in a sandboxed or constrained subprocess, executes physical rollouts, and produces the only authoritative reward report.

4 Evaluation Results

4.1 Setup and Metrics

Agents and Models We evaluate the 11 model–harness combinations listed in Table 3. The coding-agent harness provides agents with the tools to read/write files, execute bash commands, and manage contexts, and the task-specific robotics scaffoldings described above are added on top of these harnesses. Each evaluation configuration records the exact model identifier, harness version, reasoning setting, and resource budget. For Claude series, GPT series, Gemini series, and Grok series models, we use the default context management strategy configured by their own harness; for open-sourced models, including GLM-5.3-Flash and DeepSeek-V4.1-Flash, we use the same context management strategy as Claude Opus 5. Reasoning efforts are set to *high* for all combinations. Web-search tools are banned by default to prevent information leakage.

Table 3: Evaluated model-harness combinations.

Model	Harness
Claude Fable 5.1 (Anthropic, 2026a)	Claude Code
Claude Opus 5 (Anthropic, 2026c)	Claude Code
Claude Opus 4.8 (Anthropic, 2026b)	Claude Code
GPT-6 Astra (OpenAI, 2026b)	Codex CLI
GPT-5.6 Sol (OpenAI, 2026a)	Codex CLI
GPT-5.6 Terra (OpenAI, 2026a)	Codex CLI
GPT-5.6 Luna (OpenAI, 2026a)	Codex CLI
Gemini 3.7 Flash (Google DeepMind, 2026)	Antigravity CLI
Grok 4.6 (Grok, 2026)	Grok Build
GLM-5.3-Flash (Z.ai, 2026)	Claude Code
DeepSeek-V4.1-Flash (DeepSeek-AI, 2026)	Claude Code

Metrics Each task τ_i from family $\mathcal{F}_j$ and workflow $\mathcal{G}_k$ retains a native score $S_{i,j,k} \in [0, 1]$. We aggregate the score at three levels: average score across subtasks $\rightarrow$ average score across families within one workflow $\rightarrow$ average score across workflows. The final score is what we report as **RLE-Index**. Formally, for a primary family $\mathcal{F}_j$ and workflow $\mathcal{G}_k$, we define

$$S_{j,k} = \frac{1}{|\mathcal{F}_j|} \sum_{\tau_i \in \mathcal{F}_j} S_{i,j,k}, \qquad S_k = \frac{1}{|\mathcal{G}_k|} \sum_{f_j \in \mathcal{G}_k} S_{j,k}, \qquad S^{\mathrm{RLE}} = \frac{1}{4} \sum_{k=1}^{4} S_k. \tag{2}$$

The overall leaderboard is ranked by S^{RLE}. Thus, each workflow family has equal weight. We show family scores in the same table and report all native task metrics in the appendix.

4.2 Leaderboard, Capability Profiles and Cost

The overall RLE-Bench leaderboard is shown in Figure 1, with per-workflow scores compared in Figure 3.

Finding #1: GPT-6 Astra and Claude Fable 5.1 lead the leaderboard, with visual grounding emerging as a key differentiator. GPT-6 Astra and Claude Fable 5.1 outperform the other open- and closed-source models by a substantial margin. Their advantage is most pronounced in *Interactive Control* and *Perception and Estimation*, both of which require interpreting multimodal observations

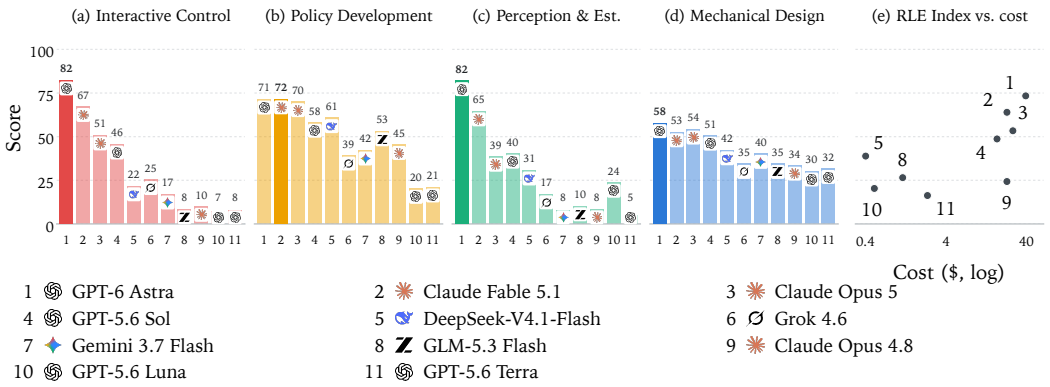


Figure 3: Capability profile per workflow (a-d) and performance versus cost (e).

and acting on repeated environment feedback. This pattern suggests that stronger visual grounding contributes substantially to their overall lead.

Finding #2: Performance is more closely matched in policy development. In *Policy Development*, which more closely resembles traditional machine learning engineering tasks, the performance gap between models is substantially smaller.

Finding #3: Understanding physical consequences remains hard for all models. On tasks from *Mechanical Design*, performance gaps are much smaller, and no model consistently makes sound design decisions, particularly when implementation choices have delayed or indirect physical consequences.

4.3 Case Studies and Key Insights

Finding #4: External robotics scaffolding generally helps, until the model is strong enough. In `Family01`, we evaluate three scaffolding levels: `L1` provides basic robot and sensor APIs; `L2`, inspired by CaP-X (Fu et al., 2026), adds SAM3 (Carion et al., 2026) and Contact-GraspNet (Sundermeyer et al., 2021) for perception and grasp planning; and `L3` adds privileged object and fixture positions. Figure 4 shows that richer scaffolding generally improves task scores and reduces costs for most models, including GPT-5.6, the Opus series, and Gemini-3.7-Flash. GPT-6-Astra, however, performs strongly with `L1`, gaining little or occasionally losing performance with additional support. These results suggest that external modules become less beneficial as model capabilities improve and may sometimes interfere with the model's own perception and reasoning.

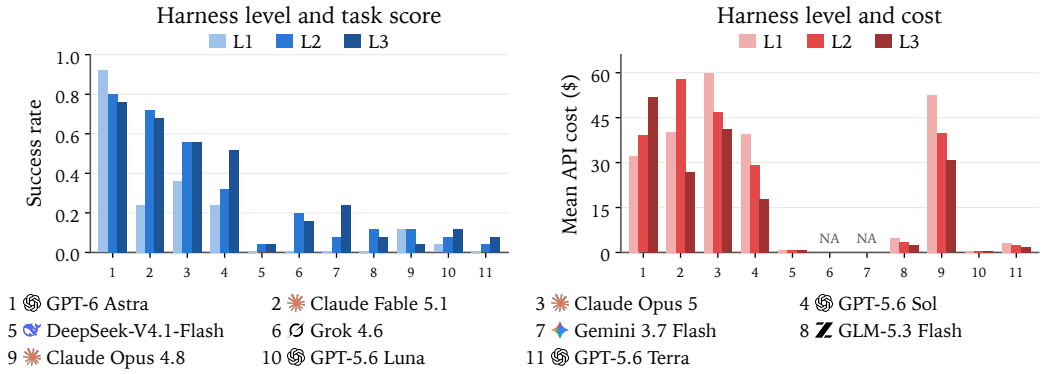


Figure 4: Effect of harness level on task success rate and cost.

Finding #5: Evaluation feedback drives real improvements within the scope it measures. In `Family05`, agents may query an evaluation service to obtain a preliminary assessment of their policy during development. We monitor and log this process, which allows us to replay the agents' progress

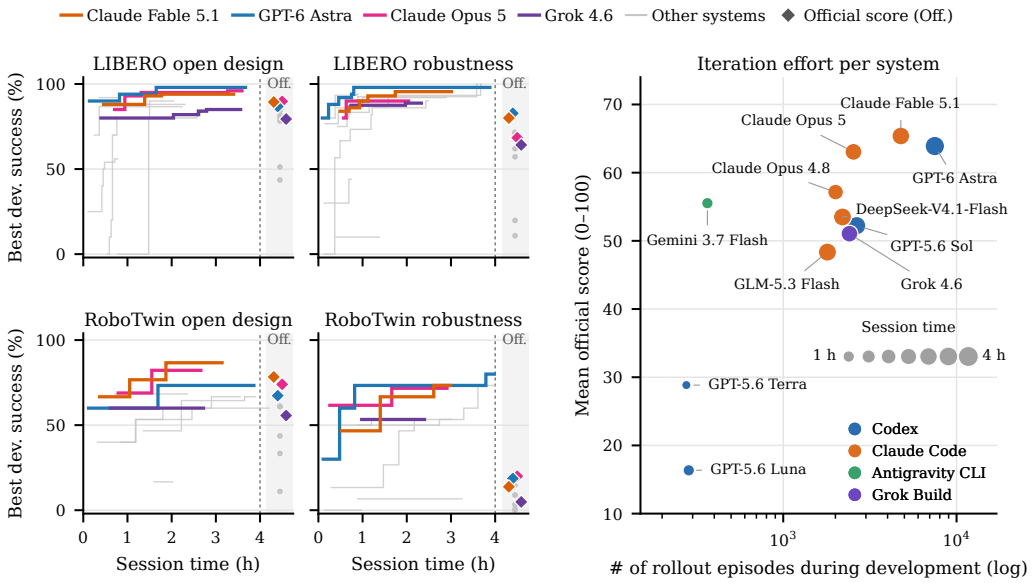


Figure 5: Development progress in the four `Family05` subtasks. **Left:** best development success so far over the four-hour session, one line per session; lines end when the agent stops, and markers in the shaded column represents the final evaluation score. **Right:** rollout episodes each system requested during development against its mean official score.

in Figure 5. Inspecting the curves from the development stage, we find that the development-time evaluation service delivers real, front-loaded improvement. In 29 of 36 sessions, the submitted version outperforms the agent's first development-time evaluation. From the right figure, we also found that the evaluation score tends to correlate positively with the rollout episodes requested during development. However, comparing final development scores against final official scores reveals that this improvement is bounded by what the feedback can measure. On open-design subtasks, where the evaluation task set is rehearsable, development success predicts the official score within 5 points. On robustness subtasks, however, whose test-time perturbations are never shown to the agent, the official score falls 19 (LIBERO) and 41 (RoboTwin) points below the development score. Several agents construct their own perturbation proxies, but none closes the gap, indicating that building robust policies remains a difficult challenge.

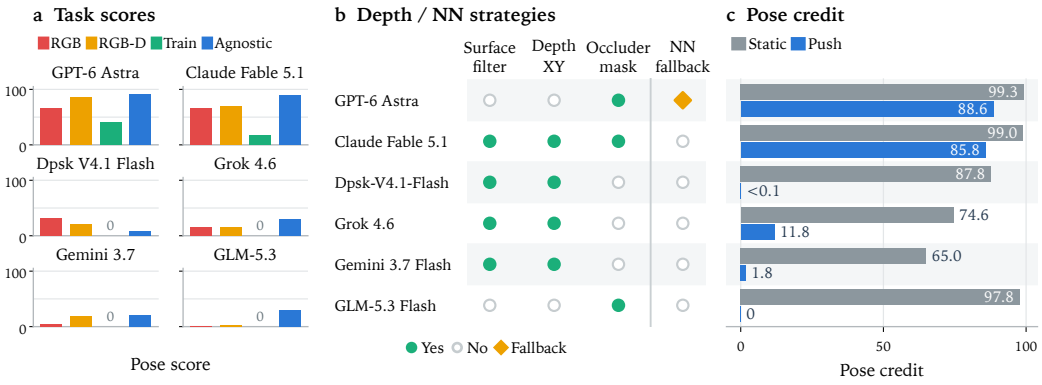


Figure 6: Behavior analysis in `Family06`, pose estimation. **Left:** scores of selected agents in different tasks in `Family06`; **Middle:** how the depth observation is used in method-agnostic task; **Right:** Score comparison between static frames and dynamic, occluded frames in method-agnostic task.

Finding #6: Depth does improve perception, while multi-sensor fusion remains difficult. Figure 6 analyzed how the depth sensor affect perception system performance in `Family06`. First, most models improve the perception score with RGB-D information compared to RGB only, showing preliminary capability to understand multiple types of physical sensors. The real gap appears when we compare different strategies for using depth information with occlusion. High-performing models use depth to judge the binary occlusion masks rather than only recover the surface or 3D positions. Notably, GPT-6 Astra trains a small convolutional neural network as a fallback when depth information is not reliable, making it the only agent that actually uses the GPU in the method-agnostic task. Final results showed that, even for tasks such as simple 2D pose estimation, only the flagship models can build a reliable perception system when occlusions get in the way.

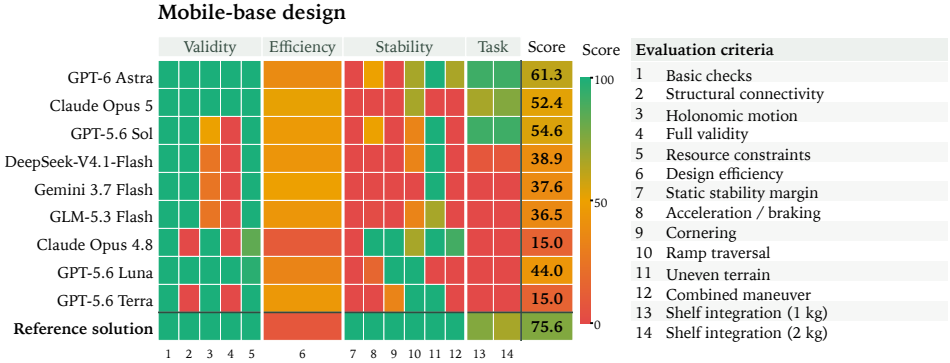


Figure 7: Sub-score of selected verifier evaluation criteria in mobile base design task results.

Finding #7: Agents can satisfy visible objectives while still missing coupled physical consequences. Figure 7 presents selected results from the `Family08` mobile base design task. Designs are evaluated for structural validity, design efficiency, static and dynamic stability, and task completion. The instructions indicate all required properties without revealing the underlying verification protocols. Most models pass the structural validity checks, demonstrating their ability to produce mechanically plausible designs. However, these designs often fail physical evaluation: all nine models receive zero worst-arm credit for static stability margin. These cases show that agents can satisfy visible design objectives while overlooking coupled physical consequences, particularly instability under load and collisions during motion.

5 Conclusion

We introduced RLE-Bench as a qualifying exam for coding agents as robot learning engineers through executable, physics-grounded tasks spanning four representative workflow families. The benchmark combines an overall leaderboard with capability profiles, trusted hidden evaluation, and explicit resource accounting. We hope RLE-Bench provides a reproducible instrument for developing agents that can build, test, and improve reliable robot-learning systems.

Although this benchmark evaluates agents exclusively in simulation, errors in agent-generated code, learned policies, or mechanical designs could have serious physical consequences if deployed in the real world. We therefore encourage responsible use of the benchmark, human oversight when interpreting its results, and independent validation before deployment.

The benchmark also has several limitations. First, simulation performance does not establish real-world reliability or safety. The scale and diversity of the simulated scenes are insufficient to assess agent behavior across all deployment conditions. Second, the four representative workflows covered in this version capture only part of the work performed by robot learning engineers; real-world robotics requires a broader range of capabilities. Third, the planned public release introduces a risk of future training data contamination, which could compromise the validity of subsequent evaluations.

AI USE STATEMENT

In this work, we used generative AI tools to assist in the writing of the paper, implement some of the tasks, and provide feedback on the task design.

We have not used generative AI tools to generate synthetic datasets, help develop theoretical models or conceptual frameworks, formulate mathematical claims, provide critical ingredients for proving mathematical claims, propose or refine hypotheses, clean and reformat datasets, support qualitative and thematic data analysis, or interpret results.

Additionally, we used generative AI tools to organize the experiment results and improve the visualization of the results.

We reviewed all AI-assisted content, including text and figures, to ensure that it accurately reflects the paper's ideas and underlying data. We take full responsibility for the final content of this work, including any text, claims, or artifacts produced with the aid of generative AI.

ETHICS STATEMENT

This benchmark evaluates coding agents on robotics tasks, where errors in generated code, learned policies, or mechanical designs may have physical consequences. Benchmark performance should not be interpreted as evidence that an agent or its artifacts are safe for real-world deployment. Deployment requires independent validation, appropriate safeguards, and human oversight, particularly around people or in safety-critical settings. Robotics capabilities may also be used for harmful purposes, including surveillance or weaponization; our intended use is to support research on the capabilities, limitations, and reliability of coding agents for robotics. We encourage responsible use of the benchmark.

REPRODUCIBILITY STATEMENT

We include the source code in the supplementary materials to enable readers to rerun our experiments. However, differences in initialization and hardware may not guarantee bit-exact reproduction of the reported results.

REFERENCES

Anthropic. Claude Fable 5.1 system card, September 2026a. URL https://www-cdn.anthropic.com/0339e6a7c5c7b87f5c07798616dc32c215d14235/Claude%20Fable%205.1%20&%20Claude%20Mythos%205.1%20System%20Card.pdf.

Anthropic. Claude Opus 4.8 system card, May 2026b. URL https://anthropic.com/claude-opus-4-8-system-card.

Anthropic. Claude Opus 5 system card, July 2026c. URL https://anthropic.com/claude-opus-5-system-card.

Kevin Black, Noah Brown, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, Lachy Groom, Karol Hausman, Brian Ichter, et al. π_0: A vision-language-action flow model for general robot control. *arXiv preprint arXiv:2410.24164*, 2024.

Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. Openai gym. *arXiv preprint arXiv:1606.01540*, 2016.

Anthony Brohan, Noah Brown, Justice Carbajal, Yevgen Chebotar, Joseph Dabis, Chelsea Finn, Keerthana Gopalakrishnan, Karol Hausman, Alex Herzog, Jasmine Hsu, et al. Rt-1: Robotics transformer for real-world control at scale. *arXiv preprint arXiv:2212.06817*, 2022.

Nicolas Carion, Laura Gustafson, Yuan-Ting Hu, Shoubhik Debnath, Ronghang Hu, Didac Suris Coll-Vinent, Chaitanya Ryali, Kalyan Vasudev Alwala, Haitham Khedr, Andrew Huang, et al. Sam 3: Segment anything with concepts. In *International conference on learning representations*, volume 2026, pp. 138846–138923, 2026.

Jun Shern Chan, Neil Chowdhury, Oliver Jaffe, James Aung, Dane Sherburn, Evan Mays, Giulio Starace, Kevin Liu, Leon Maksin, Tejal Patwardhan, et al. Mle-bench: Evaluating machine learning agents on machine learning engineering. In *International Conference on Learning Representations*, volume 2025, pp. 50466–50494, 2025.

Tianxing Chen, Zanxin Chen, Baijun Chen, Zijian Cai, Yibin Liu, Zixuan Li, Qiwei Liang, Xianliang Lin, Yiheng Ge, Zhenyu Gu, et al. Robotwin 2.0: A scalable data generator and benchmark with strong domain randomization for robust bimanual robotic manipulation. *arXiv preprint arXiv:2506.18088*, 2025.

Tianxing Chen, Yue Chen, Zixuan Li, Junyuan Tang, Kailun Su, Haoran Lu, Weijie Wan, Baijun Chen, Songling Liu, Haowen Yan, et al. Robodojo: A unified sim-and-real benchmark for comprehensive evaluation of generalist robot manipulation policies. *arXiv preprint arXiv:2607.04434*, 2026.

DeepSeek-AI. DeepSeek-V4.1-Flash. Hugging Face model card, 2026. URL `https://huggingface.co/deepseek-ai/DeepSeek-V4.1-Flash`.

Karim Elmaaroufi, Justin Svegliato, Sarunas Kalade, Graham Schelle, Sanjit A Seshia, and Matei Zaharia. Rho: Your coding agent is secretly a roboticist. *arXiv preprint arXiv:2606.16458*, 2026.

Letian Fu, Justin Yu, Karim El-Refai, Ethan Kou, Haoru Xue, Huang Huang, Wenli Xiao, Guanzhi Wang, Dantong Niu, Fei-Fei Li, et al. Cap-x: A framework for benchmarking and improving coding agents for robot manipulation. *arXiv preprint arXiv:2603.22435*, 2026.

Google DeepMind. Gemini 3.7 Flash model card, August 2026. URL `https://deepmind.google/models/model-cards/gemini-3-7-flash/`.

Grok. Grok 4.6 system card, August 2026. URL `https://media.x.ai/v1/website/card-4p6-4cd2dc57.pdf`.

Félix G. Harvey, Mike Yurick, Derek Nowrouzezahrai, and Christopher Pal. Robust motion in-betweening. *ACM Transactions on Graphics (Proceedings of ACM SIGGRAPH)*, 39(4), 2020.

Qian Huang, Jian Vora, Percy Liang, and Jure Leskovec. Mlagentbench: Evaluating language agents on machine learning experimentation. *arXiv preprint arXiv:2310.03302*, 2023a.

Wenlong Huang, Chen Wang, Ruohan Zhang, Yunzhu Li, Jiajun Wu, and Li Fei-Fei. Voxposer: Composable 3d value maps for robotic manipulation with language models. *arXiv preprint arXiv:2307.05973*, 2023b.

Stephen James, Zicong Ma, David Rovick Arrojo, and Andrew J Davison. Rlbench: The robot learning benchmark & learning environment. *IEEE Robotics and Automation Letters*, 5(2):3019–3026, 2020.

Carlos E Jimenez, John Yang, Alexander Wettig, Shunyu Yao, Kexin Pei, Ofir Press, and Karthik Narasimhan. Swe-bench: Can language models resolve real-world github issues? In *International Conference on Learning Representations*, volume 2024, pp. 54107–54157, 2024.

Yufeng Jin, Jianfei Guo, Xiaogang Jia, Yu Deng, Zechu Li, Han Liu, Weiran Liao, Vignesh Prasad, Mathias Franzius, Gerhard Neumann, et al. Nautilus: From one prompt to plug-and-play robot learning. *arXiv preprint arXiv:2605.11665*, 2026.

Jacky Liang, Wenlong Huang, Fei Xia, Peng Xu, Karol Hausman, Brian Ichter, Pete Florence, and Andy Zeng. Code as policies: Language model programs for embodied control. In *2023 IEEE International conference on robotics and automation (ICRA)*, pp. 9493–9500. IEEE, 2023.

Bo Liu, Yifeng Zhu, Chongkai Gao, Yihao Feng, Qiang Liu, Yuke Zhu, and Peter Stone. Libero: Benchmarking knowledge transfer for lifelong robot learning. *Advances in Neural Information Processing Systems*, 36:44776–44791, 2023.

Yecheng Jason Ma, William Liang, Guanzhi Wang, De-An Huang, Osbert Bastani, Dinesh Jayaraman, Yuke Zhu, Jim Fan, et al. Eureka: Human-level reward design via coding large language models. In *International conference on learning Representations*, volume 2024, pp. 26516–26560, 2024.

Oier Mees, Lukas Hermann, Erick Rosete-Beas, and Wolfram Burgard. Calvin: A benchmark for language-conditioned policy learning for long-horizon robot manipulation tasks. *IEEE Robotics and Automation Letters*, 7(3):7327–7334, 2022.

Mike Merrill, Alexander Shaw, Nicholas Carlini, Boxuan Li, Harsh Raj, Ivan Bercovich, Lin Shi, Jeong Shin, Thomas Walshe, E Kelly Buchanan, et al. Terminal-bench: Benchmarking agents on hard, realistic tasks in command line interfaces. In *International Conference on Learning Representations*, volume 2026, pp. 40903–40986, 2026.

Mayank Mittal, Pascal Roth, James Tigue, Antoine Richard, Octi Zhang, Peter Du, Antonio Serrano-Munoz, Xinjie Yao, René Zurbrügg, Nikita Rudin, et al. Isaac lab: A gpu-accelerated simulation framework for multi-modal robot learning. *arXiv preprint arXiv:2511.04831*, 2025.

Yao Mu, Tianxing Chen, Zanxin Chen, Shijia Peng, Zhiqian Lan, Zeyu Gao, Zhixuan Liang, Qiaojun Yu, Yude Zou, Mingkun Xu, et al. Robotwin: Dual-arm robot benchmark with generative digital twins. In *2025 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 27649–27660. IEEE, 2025.

Soroush Nasiriany, Abhiram Maddukuri, Lance Zhang, Adeet Parikh, Aaron Lo, Abhishek Joshi, Ajay Mandlekar, and Yuke Zhu. Robocasa: Large-scale simulation of everyday tasks for generalist robots. *arXiv preprint arXiv:2406.02523*, 2024.

OpenAI. GPT-5.6 system card, July 2026a. URL https://deploymentsafety.openai.com/gpt-5-6.

OpenAI. GPT-6 Astra system card, September 2026b. URL https://deploymentsafety.openai.com/gpt-6-astra.

Rushi Qiang, Yuchen Zhuang, Yinghao Li, Dingu Sagar V K, Rongzhi Zhang, ChangHao Li, Ian Wong, Sherry Yang, Percy Liang, Chao Zhang, and Bo Dai. Mle-dojo: Interactive environments for empowering llm agents in machine learning engineering. In D. Belgrave, C. Zhang, H. Lin, R. Pascanu, P. Koniusz, M. Ghassemi, and N. Chen (eds.), *Advances in Neural Information Processing Systems*, volume 38, Main Conference. Curran Associates, Inc., 2025. doi: 10.52202/085713-0139. URL https://proceedings.neurips.cc/paper_files/paper/2025/file/0603c69125ad4b964bc9c4832f7b9f8f-Paper-Datasets_and_Benchmarks_Track.pdf.

Martin Sundermeyer, Arsalan Mousavian, Rudolph Triebel, and Dieter Fox. Contact-graspnet: Efficient 6-dof grasp generation in cluttered scenes. In *2021 IEEE international conference on robotics and automation (ICRA)*, pp. 13438–13444. IEEE, 2021.

Stone Tao, Fanbo Xiang, Arth Shukla, Yuzhe Qin, Xander Hinrichsen, Xiaodi Yuan, Chen Bao, Xinsong Lin, Yulin Liu, Tse-kai Chan, et al. Maniskill3: Gpu parallelized robotics simulation and rendering for generalizable embodied ai. *arXiv preprint arXiv:2410.00425*, 2024.

Yuval Tassa, Yotam Doron, Alistair Muldal, Tom Erez, Yazhe Li, Diego de Las Casas, David Budden, Abbas Abdolmaleki, Josh Merel, Andrew Lefrancq, et al. Deepmind control suite. *arXiv preprint arXiv:1801.00690*, 2018.

Harbor Framework Team. Harbor: A framework for evaluating and optimizing agents and models in container environments. *Zenodo*, 2026.

Philipp Wu, Yide Shentu, Zhongke Yi, Xingyu Lin, and Pieter Abbeel. Gello: A general, low-cost, and intuitive teleoperation framework for robot manipulators. In *2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 12156–12163. IEEE, 2024.

Wenli Xiao, Jia Xie, Tonghe Zhang, Haotian Lin, Letian Fu, Haoru Xue, Jalen Lu, Yi Yang, Cunxi Dai, Zi Wang, et al. Enpire: Agentic robot policy self-improvement in the real world. *arXiv preprint arXiv:2606.19980*, 2026.

Tianbao Xie, Danyang Zhang, Jixuan Chen, Xiaochuan Li, Siheng Zhao, Ruisheng Cao, Toh J Hua, Zhoujun Cheng, Dongchan Shin, Fangyu Lei, et al. Osworld: Benchmarking multimodal agents for open-ended tasks in real computer environments. *Advances in Neural Information Processing Systems*, 37:52040–52094, 2024a.

Tianbao Xie, Siheng Zhao, Chen Wu, Yitao Liu, Qian Luo, Victor Zhong, Yanchao Yang, and Tao Yu. Text2reward: Reward shaping with language models for reinforcement learning. In *International Conference on Learning Representations*, volume 2024, pp. 35663–35699, 2024b.

Seonghyeon Ye, Yunhao Ge, Kaiyuan Zheng, Shenyuan Gao, Sihyun Yu, George Kurian, Suneel Indupuru, You Liang Tan, Chuning Zhu, Jiannan Xiang, et al. World action models are zero-shot policies. *arXiv preprint arXiv:2602.15922*, 2026.

Z.ai. GLM-5.3-Flash. Hugging Face model card, 2026. URL `https://huggingface.co/zai-org/GLM-5.3-Flash`.

Yuke Zhu, Josiah Wong, Ajay Mandlekar, Roberto Martín-Martín, Abhishek Joshi, Kevin Lin, Abhiram Maddukuri, Soroush Nasiriany, and Yifeng Zhu. robosuite: A modular simulation framework and benchmark for robot learning. *arXiv preprint arXiv:2009.12293*, 2020.

A Complete Task Cards

Task Family 01: RoboCasa Speed-Run

15 instances

Family: Interactive control
Environment: RoboCasa
Task matrix: Five kitchen tasks × three harness levels

AGENT INSTRUCTIONS (PARTIAL)

You are controlling a simulated Franka-class arm on a mobile base in a RoboCasa kitchen. The task is **OpenFridge**—open the refrigerator door. Your goal in this phase is to **learn to solve it and write the code you will need**. [. . .] Evaluation will use the same task in unseen kitchen layouts.

ENVIRONMENT / EXECUTION VIEW



Example: OpenFridge. Three camera views from the official RoboCasa demonstration (illustrative).

Develop on `pretrain` kitchens, then evaluate in five distinct `target` kitchens. Files persist; conversation resumes when supported.

Five task variants

`OpenFridge`	Open the refrigerator door.
`CloseCabinet`	Close the cabinet door.
`TurnOnStove`	Turn on the specified burner.
`PickPlaceCounterToDrawer`	Move the target into the drawer.
`PickPlaceMicrowaveToCounter`	Move the target to its counter destination.

PROVIDED HARNESS

- Task instruction, simulator client, and a persistent coding workspace.
- **L1:** Three RGB cameras, optional metric depth, and robot proprioception.
- **L2 adds:** Camera geometry, frame transforms, point-cloud utilities, motion primitives, SAM3 segmentation, and Contact-GraspNet grasp proposals.
- **L3 adds:** World-frame object and fixture poses, fixture extents, target identity, and grasp-related state.

SUBMITTED ARTIFACTS

- Controller and utility code developed in the workspace and used during evaluation.
- Closed-loop 12-D actions in $[-1, 1]$: arm (6), gripper (1), mobile base (3), torso (1), and mode (1).
- Physical task completion, confirmed by the simulator's success predicate.

EVALUATION RUBRIC

Let s be successful trials divided by five, and d be development steps used:

$$R = s\left[0.80 + 0.20\left(1 - \frac{d}{50{,}000}\right)\right].$$

- Success determines 80% of reward; interaction efficiency contributes up to 20%, scaled by success.
- Missing trials count as failures. A missing, unsealed, or unverifiable ledger scores zero.

BUDGET / EVALUATION PROTOCOL

- **Develop:** 8 hours, 1 GPU, and 50,000 charged steps. Each reset and executed action costs one step; observations are free.
- **Evaluate:** One terminal phase, 1 hour total; five trials with at most 1,000 actions each.
- Evaluation actions do not enter d. Reset advances to the next trial.
- Library-generated actions are metered, and the trusted ledger records interaction and success.

Task Family 02: RoboCasa Harness Transfer

15 instances

Family: Interactive control
Environment: RoboCasa
Task matrix: Five easy, five medium, and five hard transfer groups

AGENT INSTRUCTIONS (PARTIAL)

You are controlling a simulated Franka-class arm on a mobile base in a RoboCasa kitchen. [. . .] Your job is a **harness**: perception primitives, controllers built on them, and a manual. When this phase ends, a **different agent** [. . .] gets one shot at one kitchen task using only what you left behind. Your score is entirely what that agent achieves.

ENVIRONMENT / EXECUTION VIEW



Example: Washing Dishes. The held-out task is `DumpLeftovers`, illustrated by an official RoboCasa demo.

Develop on	`PlaceOnDishRack`, `SortingCleanup`, `StackBowlsInSink`.
Transfer	Only `agent_harness/` code and manual reach each fresh agent.
Evaluate	Five fresh agents independently attempt `DumpLeftovers`.

PROVIDED HARNESS

- Three selected composite tasks per activity group; their atomic skills cover the held-out task.
- Three RGB views, optional metric depth, robot proprioception, and a metered simulator client.
- Observation-only perception: object and fixture poses are withheld.

SUBMITTED ARTIFACTS

- A reusable `agent_harness/` with perception functions, closed-loop controllers, and a usage manual.
- No development conversation or other workspace files are transferred.
- Evaluation agents use the harness to execute 12-D robot actions.

EVALUATION RUBRIC

For baseline, peak simultaneous, and total satisfied conjunct counts b, m, n:

$$s_{\text{trial}} = \text{clip}_{[0,1]}\left(\frac{m-b}{n-b}\right).$$

- Simulator success gives 1. Otherwise, missing stage records or $n = b$ give 0.
- Reward is the mean over five agents; missing trials count as zero.
- Binary success is reported separately. Development efficiency adds no reward.

BUDGET / EVALUATION PROTOCOL

- **Develop:** 8 h, 1 GPU, 75,000 charged steps; resets and executed actions each cost one step.
- **Evaluate:** Five fresh agents, each with 1 h and 5,000 steps.
- Observations are free. Evaluation uses unseen kitchens and the held-out task.

Task Family 03: Tabletop Physical Reasoning

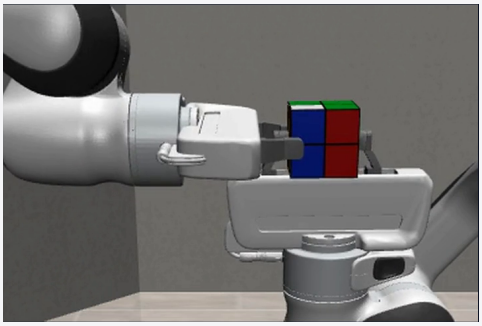
5 instances

Family: Interactive control
Environment: MuJoCo / robosuite
Task matrix: Construction, physical inference, and contact manipulation

AGENT INSTRUCTIONS (PARTIAL)

Write a controller that solves a scrambled 64 mm $2 \times 2 \times 2$ Rubik's Cube using two fixed Panda arms. [. . .] Grasp opposing halves and rotate one half relative to the other. Turn layers through gripper contact and reorient the cube as needed to inspect hidden faces. Finish with all six faces solved and aligned, and the cube held still above the table.

ENVIRONMENT / EXECUTION VIEW



Example: Pocket cube. Two Panda grippers exchange the cube during a development attempt. Frame from a GPT-6 Astra evaluation trajectory, showing a retry after a missed handoff.

Tower	Build a stable, unsupported tower from 12 mixed pieces.
Cantilever	Maximize the overhang of four supported blocks.
Balance	Identify the heavy cube among nine and place it on the mat.
Pocket cube	Solve a physical $2 \times 2 \times 2$ cube with two arms.
Hidden COM	Identify the ballast quadrant in each of three sealed boxes.

PROVIDED HARNESS

- Tower / cantilever / balance: three RGB views, proprioception, 12-D actions, and a Cartesian movement helper.
- Balance also exposes cube and pan positions; masses remain hidden.
- Cube: one front RGB camera, optional depth, two-arm state, and 14-D actions.
- Hidden COM: three RGB views and robot state; no force, contact, mass, or object-pose readings.

SUBMITTED ARTIFACTS

- Tower / cantilever / balance: the final physical scene.
- Cube: a simulator-confirmed physical solve, with efficient face turns.
- Hidden COM: one irreversible A/B/C/D answer per box; no correctness feedback.

EVALUATION RUBRIC

- Tower / cantilever: settled height or overhang divided by the analytic optimum, clipped to $[0, 1]$.
- Balance: zero unless only the heavy cube reaches the mat; then 1 for $w \leq 2$ weighings, or $2/w$.
- Cube: solve required; excess quarter turns reduce reward relative to the optimum. Invalid counters give zero; unclassified transitions cap reward at 0.5.
- Hidden COM: fraction of three answers correct.

BUDGET / EVALUATION PROTOCOL

- Tower / cantilever / balance / cube: 9 h, 1 GPU, 50,000 steps; one continuous attempt.
- Resets restore the same scene for the first three tasks. Cube reset abandons the attempt; metered drop rescue preserves completed turns.
- Hidden COM: 1 h total, 12,000 steps per box. Observations are free.

Task Family 04: Whole-Body Motion Tracking

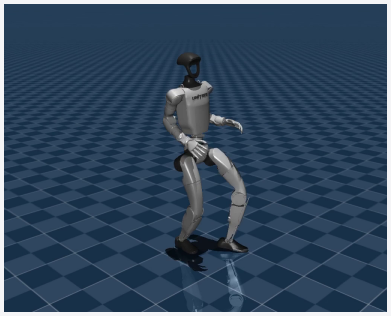
5 instances

Family: Policy development
Environment: MuJoCo-Warp / MuJoCo-C
Task matrix: Dance, fight, fall and get up, run, sprint

AGENT INSTRUCTIONS (PARTIAL)

A Unitree G1 has to reproduce a 20-second excerpt of LAFAN1 `dance1_subject2`: 29 joints, 50 Hz, matching the reference whole-body motion and world-frame trajectory without falling over. **The environment and a reference PPO are given. What you train is your choice.** [. . .] Stage early, stage often.

ENVIRONMENT / EXECUTION VIEW



Example: Dance tracking. Unitree G1 executing a learned motion-tracking policy in simulation. Frame from a GPT-6 Astra evaluation trajectory.

Workflow: train in GPU-parallel MuJoCo-Warp, export ONNX, then evaluate in CPU MuJoCo-C.

Shared target	A 20-second reference clip for each of five motions.
Control interface	160-D observation → 29 joint-position offsets at 50 Hz.

PROVIDED HARNESS

- Retargeted LAFAN1 motion, G1 model, modifiable training environment, export tools, and design-seed evaluation.
- An optional one-hour PPO baseline.
- Observations include reference motion, torso-frame anchor, robot state, and previous action.

SUBMITTED ARTIFACTS

- `policy.onnx` and `policy_meta.json`.
- At most 10 million parameters; a history of 1–16 observation frames.
- The verifier runs the exported graph directly; no agent training code is imported.

EVALUATION RUBRIC

Per hidden seed, score is clipped to $[0, 1]$:

$$s = 0.7T + 0.3U - 0.15\min(J/J_{\max}, 1) - 0.05F.$$

- T: whole-body tracking; U: survival fraction; J: action jerk; F: fall indicator.
- Reward is the mean over eight hidden seeds. Invalid exports score zero; stability or determinism smoke failure caps reward at 0.1.
- Dance thresholds are calibrated; the other four sets remain provisional.

BUDGET / EVALUATION PROTOCOL

- **Develop:** 4 h, 1 GPU; training and evaluation share the session budget.
- Train at a 5 ms physics step; evaluate at 2 ms.
- Hidden evaluation perturbs dynamics, sensors, terrain, latency, and pushes.

Task Family 05: nanoVLA Recipe Engineering

4 instances

Family: Policy development **Environment:** LIBERO / RoboTwin 2.0
Task matrix: Two environments × open-design and robustness tracks

AGENT INSTRUCTIONS (PARTIAL)

Maximise LIBERO-10 success with DINOv2-base as the only pretrained component. You get demonstrations, the encoder and a socket protocol; you decide everything else: whether to freeze, fine-tune or partially reuse the encoder, how language is handled, the policy architecture, the optimiser, the serving code. [. . .] Both modes run offline. Keep every stochastic operation seeded.

ENVIRONMENT / EXECUTION VIEW

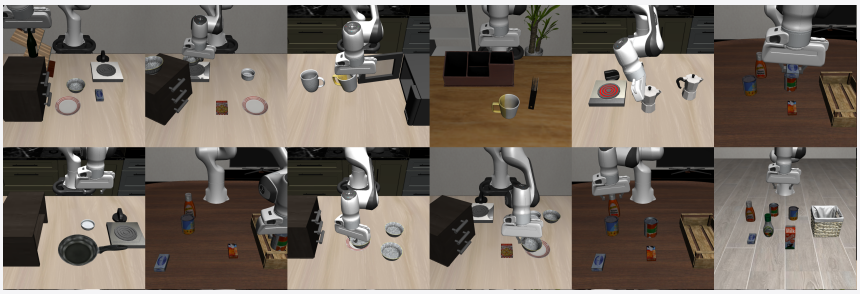


Illustration: Representative LIBERO scenes, cropped from the official project collage. RLE-Bench uses LIBERO-10 and RoboTwin tracks. Only the submitted Python recipe is transferred and retrained.

LIBERO open / robust	500 evaluation episodes each; at least 475 complete.
RoboTwin open / robust	300 / 225 episodes; at least 285 / 214 complete.

PROVIDED HARNESS

- Read-only demonstrations, approved pretrained encoders and loaders, a policy socket protocol, and public evaluation client.
- Open-design tracks allow DINOv2-base; robustness tracks provide a vision/text encoder bundle.
- No policy, trainer, simulator, or evaluator implementation is supplied.

SUBMITTED ARTIFACTS

- One UTF-8 `solution.py`, at most 1 MiB, supporting training and serving.
- Replay must write `ckpt.pt`; `meta.json` is optional. Development checkpoints are not transferred.
- Inference consumes language, two 128 × 128 RGB views, and robot state; returns chunks of 1–64 actions.

EVALUATION RUBRIC

- Reward is success rate over the fixed episode set; missing episodes are failures.
- Open tracks use official LIBERO-10 states or held-out clean RoboTwin scenes. Robust tracks use perturbations.
- Failed replay, artifact validation, or completion thresholds yield zero.
- No standard-performance gate for robustness tracks; no efficiency bonus.

BUDGET / EVALUATION PROTOCOL

- **Develop:** 4 h, 1 H100, 48 CPUs, 64 GiB; up to 20,000 requested development episodes.
- **Replay:** 1,800 s plus 120 s infrastructure allowance; training runs once offline.
- LIBERO: 8-D state / 7-D action. RoboTwin: 16-D state / 14-D joint targets. Serving is separately time-limited.

Task Family 06: Blind Planar Pose Estimation

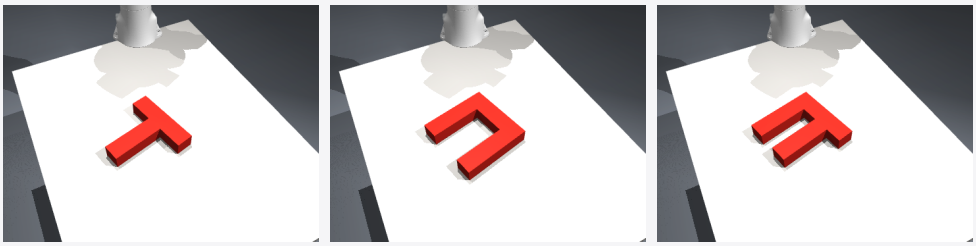
4 instances

Family: Perception and estimation
Environment: MuJoCo
Task matrix: RGB only, RGB-D, learned RGB-D, method agnostic

AGENT INSTRUCTIONS (PARTIAL)

A robot-mounted stick pushes one of three red, asymmetric blocks on a table. From a fixed oblique camera, **estimate the block's planar pose and identify its shape**. Predictions should remain accurate through rotation and occlusion, with efficient CPU inference. [. . .] Meshes and the private scene are not available; learn shape geometry and orientation conventions from rendered examples.

ENVIRONMENT / EXECUTION VIEW



Task-scene renders: T, C, and F blocks (left to right), viewed by the fixed oblique sensor camera. These geometry illustrations are not estimator predictions.

A / B	RGB-only / RGB-D Python estimators; 4 CPUs.
C	RGB-D model trained from scratch; 16 CPUs, 1 GPU.
D	Method-agnostic Python estimator; 4 CPUs, 1 GPU.

PROVIDED HARNESS

- Public deterministic renderers for individual frames and push episodes.
- Noisy pose labels, camera calibration, time, and RGB; depth is variant-dependent. Meshes and shape labels are withheld.
- Up to six design-set evaluation calls. All final inference runs on CPU.

SUBMITTED ARTIFACTS

- A / B / D: `estimator.py` with causal `reset` and `update`; return finite $(x, y, \theta, \text{shape_id})$.
- C: CPU TorchScript `model.pt`, at most 20M tensor elements; independent per-frame predictions from RGB, depth, and validity mask.
- Shape IDs 0, 1, 2 denote T, C, F.

EVALUATION RUBRIC

Each group uses the five largest translation and wrapped-angle errors, averaged separately. If all shape IDs are correct:

$$g = \tfrac{1}{2}e^{-(\bar{e}_{xy}/0.01)^2} + \tfrac{1}{2}e^{-(\bar{e}_{\theta}/0.1)^2}.$$

Otherwise $g = 0$. Overall reward is

$$R = \max(0,\ 0.3\bar{g}_A + 0.7\bar{g}_B - D).$$

$D = 0.2\,\text{clip}_{[0,1]}[(10 - \nu)/9]$, where ν is verifier-measured frames/s. A failed load or interface gate scores zero.

BUDGET / EVALUATION PROTOCOL

- **Develop:** 2 h; resources vary by track as listed above.
- **Evaluate:** 100 independent frames in ten groups, then ten push trajectories of at most 60 frames.
- Python estimators may retain causal state. The learned model is stateless.

Task Family 07: Contact-Rich Bin Clearing

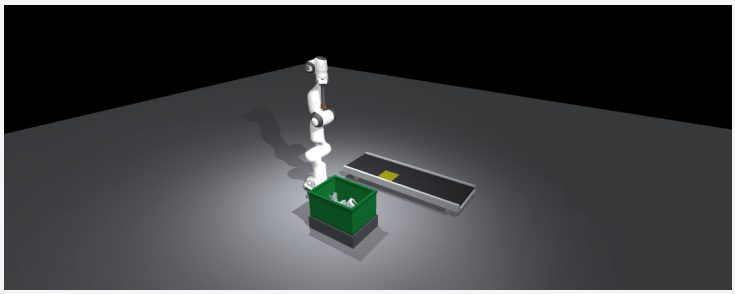
1 instance

Family: Perception and estimation
Environment: MuJoCo
Task matrix: One policy evaluated on eight hidden piles

AGENT INSTRUCTIONS (PARTIAL)

A green KLT bin arrives at your robot cell holding 12–16 stamped steel U-brackets, dumped loose. A conveyor runs along the other side of the cell. Your job: ship the software that empties the bin onto the conveyor—**as many parts as possible, as fast as possible, breaking nothing**. [. . .] The drop zone is a small placement nest [. . .] and it takes ONE part at a time.

ENVIRONMENT / EXECUTION VIEW



Task-scene render: A public pile (seed 101), contact electromagnet, and conveyor drop zone. Contact is required to pick up a bracket; delivering several together blocks the nest until extras are removed.

PROVIDED HARNESS

- Panda workcell, a contact electromagnet, public geometry and sensor calibration, and five public pile seeds.
- 20 Hz joint positions, velocities, torques, wrist force/torque, magnet state, and time.
- 10 Hz overhead and wrist RGB-D images. Ground-truth part poses are withheld.
- A development runner, diagnostic events, and optional videos.

SUBMITTED ARTIFACTS

- An isolated `policy/` package exposing `make_policy()`, `reset()`, and `act()`.
- At each control tick: seven joint-position targets and one magnet command.
- Clear brackets into the conveyor nest one at a time while avoiding floor drops, damage, and hard bin impacts.

EVALUATION RUBRIC

For cleared fraction f, throughput p, and clean complete-clear indicator I:

$$\begin{aligned} s = {} & 0.30I + 0.35h(f) \\ & + 0.20\min(p/10, 1) \\ & + 0.15I\min(p/15, 1) \\ & - 0.03N_{\text{floor}} - 0.05N_{\text{damage}} \\ & - 0.05N_{\text{bin}}. \end{aligned}$$

$h(f) = f/6$ for $f \le 0.6$; otherwise $h(f) = 0.1 + 0.9[(f - 0.6)/0.4]^3$. Clip episode scores to $[0, 1]$ and average over eight piles. Missing/unloadable policies score zero; subsequent smoke failure caps reward at 0.1.

BUDGET / EVALUATION PROTOCOL

- **Develop:** 4 h, 4 CPUs.
- **Evaluate:** Eight hidden piles, each with 120 s of simulated time and 360 s cumulative policy computation.
- CPU-only inference; each call has a 30 s hang cap. Throughput uses clear makespan, or 120 s if incomplete.

Task Family 08: Universal Mobile-Manipulator Base

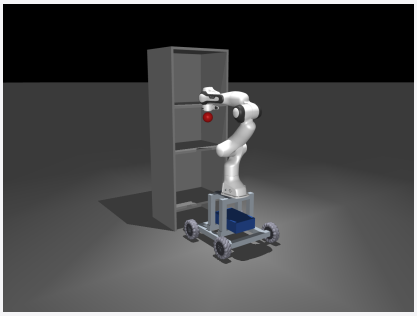
1 instance

Family: Mechanical design
Environment: MuJoCo
Task matrix: One chassis × three arms × 12 targets × two loads

AGENT INSTRUCTIONS (PARTIAL)

Design and build one mecanum mobile-manipulator base compatible with the three supplied canonical arms: Franka Panda, Universal Robots UR5e, and UFACTORY xArm7. **The same submitted chassis, battery, wheels, and `arm_mount_site` must be used unchanged with every arm.** [...] Submit `controller.py` to control shelf entry and loaded target holding.

ENVIRONMENT / EXECUTION VIEW



Example assembly. Existing project visualization of a mobile base, Panda arm, payload, and shelf.

The verifier replaces the preview arm with each canonical model. The base and mounting interface remain unchanged.

Required load: 1 kg.
Margin test: 2 kg.
All 12 shelf targets count for each arm.

PROVIDED HARNESS

- Canonical Panda (7-DoF), UR5e (6-DoF), and xArm7 (7-DoF) models.
- A fixed universal adapter, immutable mecanum wheels, battery, payload, aluminum-profile catalog, and shelf specification.
- A preview scene for testing the submitted base and controller.

SUBMITTED ARTIFACTS

- `robot.xml` and `controller.py`, with required model names and mounting site.
- At 50 Hz, the controller maps time, joint state, and current controls to one finite command per actuator.
- The submitted controller performs shelf entry and holding; verifier controllers run other design tests.

EVALUATION RUBRIC

Take the **minimum across the three arms at each checkpoint**, then apply weights:

Validity	15%
Design / resources	35%
Static stability	15%
Dynamic stability	20%
Shelf integration	15%

Shelf integration assigns 10% to 1 kg and 5% to 2 kg trials. A failed load or structure checkpoint caps reward at 0.15.

BUDGET / EVALUATION PROTOCOL

- **Develop:** 2 h, 4 CPUs.
- Three trusted arm assemblies; all 12 targets at both loads, plus static and dynamic batteries.
- Compactness, mass, and material use affect design credit. Reach and hold without shelf collision, tipping, or sustained loss of wheel support.

Task Family 09: Multi-Robot GELLO Co-Design

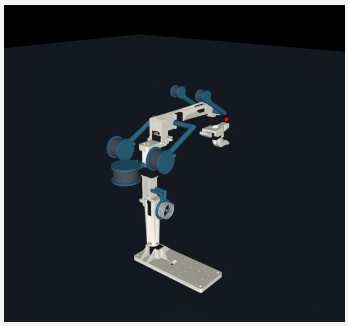
1 instance

Family: Mechanical design
Environment: MuJoCo
Task matrix: Three robot-matched leader devices, developed together

AGENT INSTRUCTIONS (PARTIAL)

Design gravity compensation for three CAD-assembled GELLO lead arms, one for each of these follower robots: Franka Panda, Universal Robots UR5e, and UFACTORY xArm7. [...] Modify `lead.xml` with springs, masses, supports, and counterweights so the passive arm holds pose while remaining easy to move. [...] Implement `trim.py` with model-based feedforward and per-device adaptation from probe measurements.

ENVIRONMENT / EXECUTION VIEW



Example: Franka GELLO. A submitted gravity-compensation mechanism in simulation. The frame shows the submitted leader arm and counterweights.

Variants: Franka (7 joints), UR5e (6), and xArm7 (7).

PROVIDED HARNESS

- Measured geometry, uncompensated leader assemblies, workspace grids, nominal dynamics utilities, and public analysis helpers.
- Preserve joint frames, geometry, mounts, and stock servos; obey mass and printability constraints.
- Hidden device copies vary link masses and handle payload.

SUBMITTED ARTIFACTS

- For each robot: modified `lead.xml`, `trim.py`, and referenced meshes.
- Passive compensation plus nominal and adapted feedforward torque functions.
- `plan_probe` selects 15 additional poses after a home measurement; `adapt` receives all 16 settled-pose probes.

EVALUATION RUBRIC

Each variant has ten equally weighted reward checkpoints:

- Motion clearance: 10%.
- Passive residual torque, hold, and backdrive: 30%.
- Adapted reference-arm hold and backdrive: 20%.
- Combined hold, backdrive, servo headroom, and push recovery: 40%.

Validity shortfalls deduct up to 0.10; floor at zero. Load/structure failure skips the submitted-arm battery and unavailable metrics score zero. Final reward averages all three variants.

BUDGET / EVALUATION PROTOCOL

- **Develop:** 3 h, 4 CPUs for all three designs.
- Maximum device mass: 2.30 kg; stock servo torque limit: 0.35 Nm.
- Probe poses are selected as one batch. The verifier freezes sampled feedforward during each scenario and tests submitted and reference assemblies.

B Full Per-Task Results

B.1 Coverage and Reading Guide

We report all 51 tasks across nine task families and four workflows for the 11 model–harness systems. All scores below are verifier rewards multiplied by 100, not uniformly success percentages. Family scores average their constituent tasks. The RLE Index first averages family scores within each workflow (01–03, 04–05, 06–07, and 08–09), then averages the four workflow scores. Every plot uses the same descending RLE Index order. Numeric annotations are rounded to one decimal; the archived export and accompanying CSV preserve the exported precision and all native metrics. Missing values are marked NA or a dash, distinct from measured zeros.

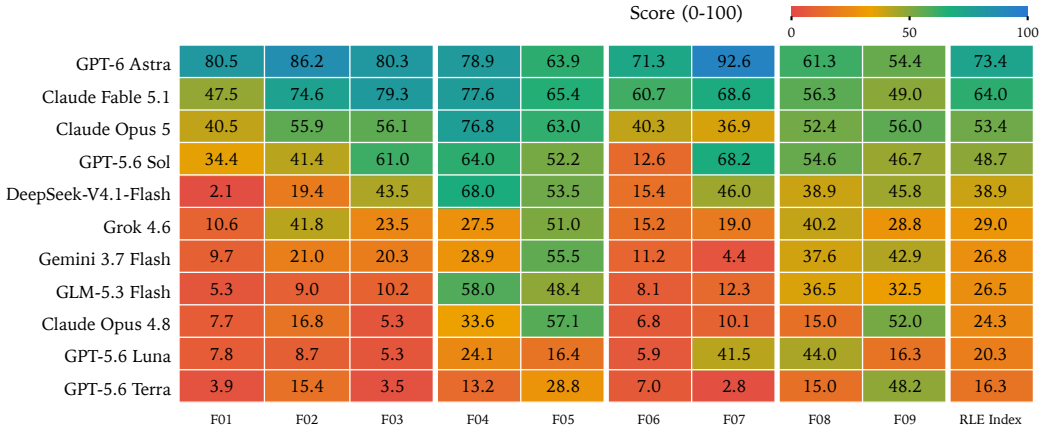


Figure 8: Task-family scores (F01–F09) and overall RLE Index for every system. The same 0–100 scale is used throughout the reward figures. The Index gives workflows equal weight, so it is not the unweighted mean of these nine columns.

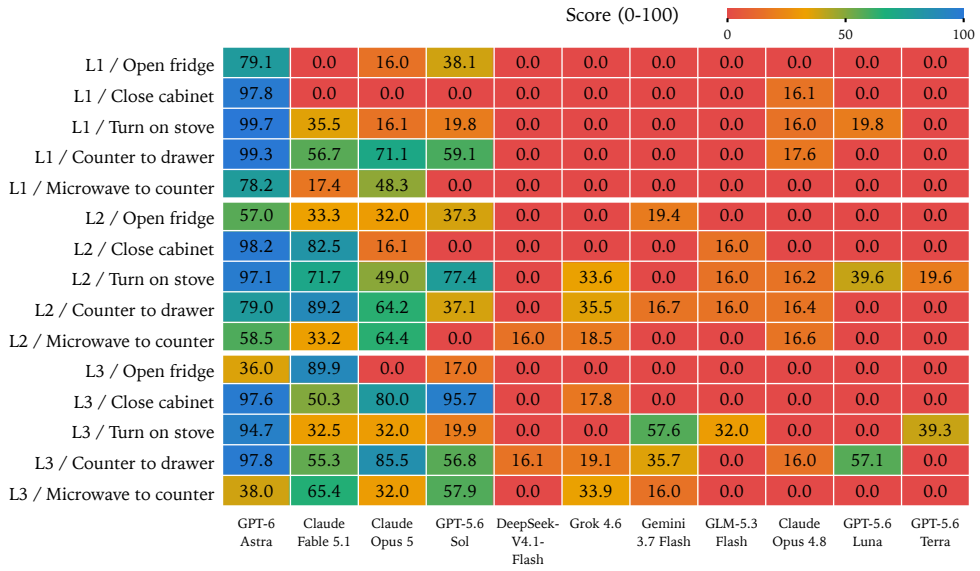


Figure 9: **Family 01: agentic control.** All 15 tasks: five kitchen activities at each of three scaffolding levels. L1 provides the action API, L2 adds the harness library, and L3 adds privileged state. Each cell is one selected run's verifier reward; horizontal separators distinguish levels.

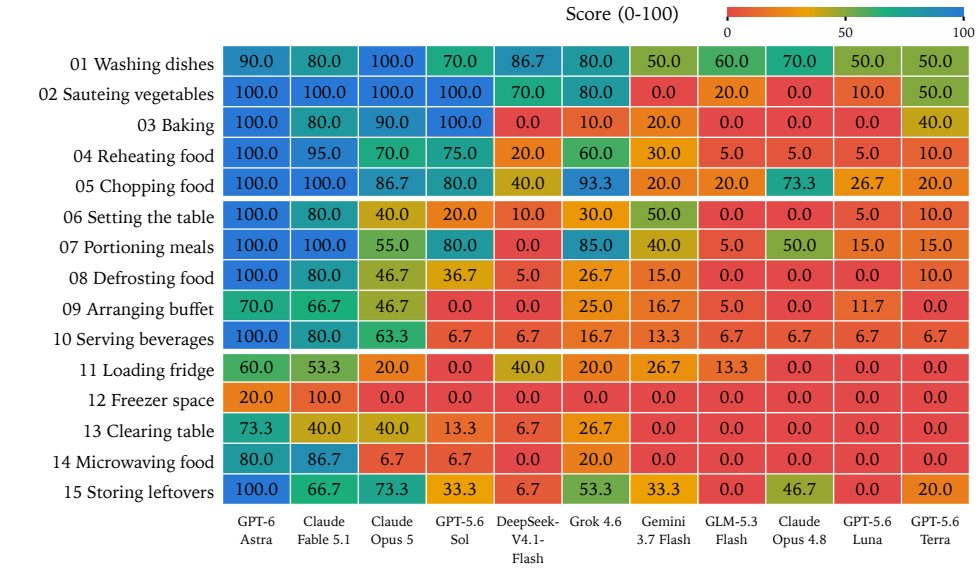


Figure 10: **Family 02: harness transfer.** All 15 activity groups, with separators after the five easy and five medium groups; the final five are hard. Each reward measures stage credit achieved by fresh agents on the group's held-out task using the submitted harness. Group numbers match the task catalog.

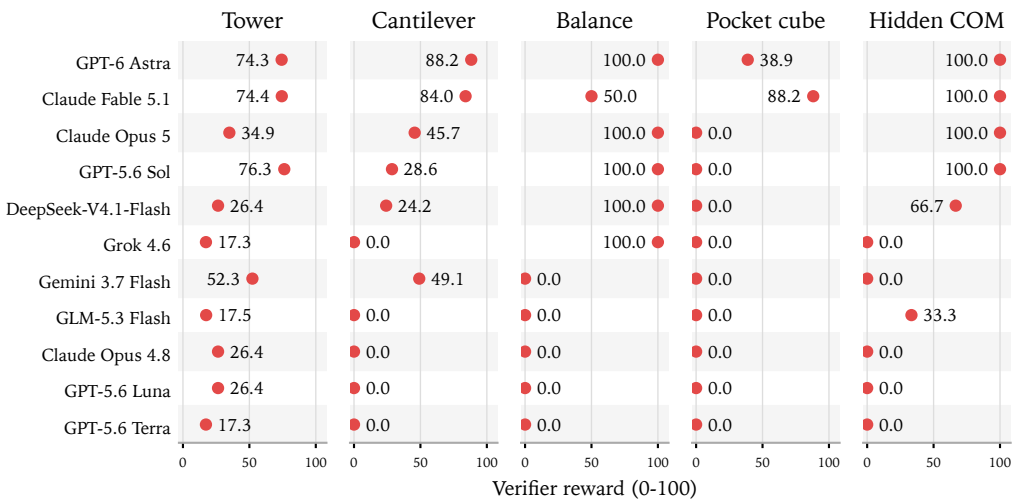


Figure 11: **Family 03: tabletop physical reasoning.** Dot plots show all five task rewards separately, with numeric annotations. The balance task uses the task identifier `03-balance-coins`; its audited task card specifies heavy-cube identification. Hidden COM denotes hidden center of mass. Values use the task-specific verifier reward, not a common physical unit.

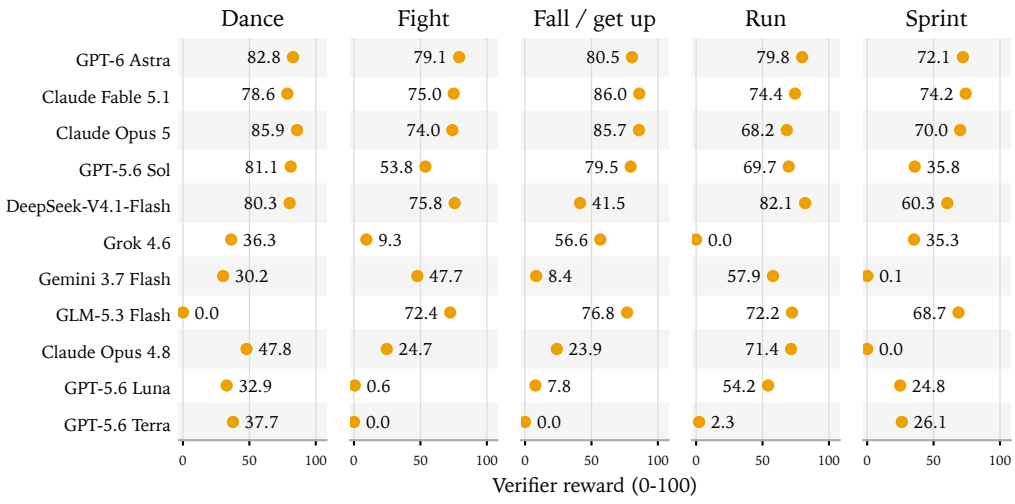


Figure 12: **Family 04: whole-body motion tracking.** Rewards for all five motion clips. Each dot represents the selected run's final verifier reward, which combines tracking and survival. Shared axes expose differences across clips without averaging them away. Native MPJPE and survival are reported in Table 4.

The two families retain distinct evaluation contracts. Family 03 scores physical outcomes or submitted answers within an interaction attempt; Family 04 evaluates a submitted tracking policy on hidden seeds. Dot plots place each task on its own axis while retaining the common normalized reward scale.

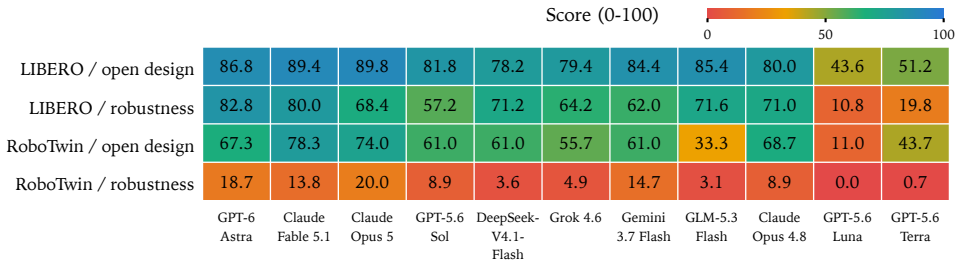


Figure 13: **Family 05: nanoVLA recipe engineering.** All four LIBERO and RoboTwin tracks, separating open design from robustness. Entries are final hidden-evaluation rewards, not the best development-time evaluation scores.

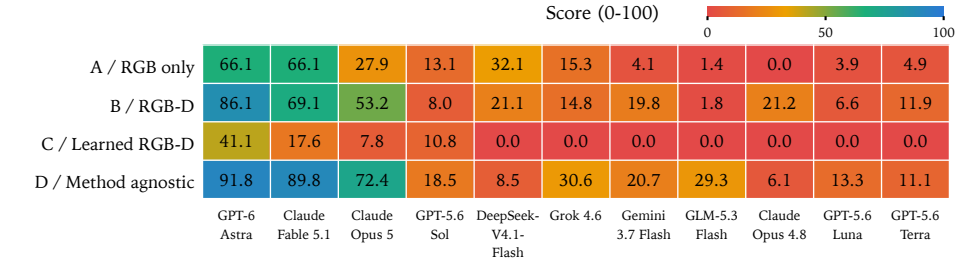


Figure 14: **Family 06: blind planar pose estimation.** All four estimator variants. A–D match the task card: RGB-only, RGB-D, learned RGB-D, and method-agnostic estimation. Entries are final rewards after the verifier inference-speed deduction and load/interface gate; they are not raw pose accuracy.

Table 4: Native diagnostics. MPJPE (lower is better) and survival are arithmetic means over the five motion clips in Family 04. Clearance and throughput are reported for Family 07. Higher is better for the latter three columns; these quantities are not interchangeable with verifier reward.

System	MPJPE (mm)	Survival (%)	Clearance (%)	Parts/min
GPT-6 Astra	32.62	98.71	100.00	9.22
Claude Fable 5.1	40.42	100.00	86.90	14.58
Claude Opus 5	34.90	93.39	78.69	6.02
GPT-5.6 Sol	51.50	87.03	96.69	8.35
DeepSeek-V4.1-Flash	43.52	91.41	87.09	6.82
Grok 4.6	62.22	44.77	71.52	5.89
Gemini 3.7 Flash	62.48	46.41	27.32	10.62
GLM-5.3 Flash	55.64	77.94	61.17	6.02
Claude Opus 4.8	69.82	51.27	52.74	3.92
GPT-5.6 Luna	79.44	41.43	90.70	6.80
GPT-5.6 Terra	89.38	26.54	29.33	4.08

Interpreting native diagnostics. Table 4 supplements normalized rewards with physical units. Motion errors and survival are averaged over clips, whereas bin-clearance and throughput are copied from the joint task report. These summaries do not reconstruct gated rewards. The machine-readable records retain every exported per-task metric, including pose errors, inference rates, gate fields, and mechanical-design checkpoints.

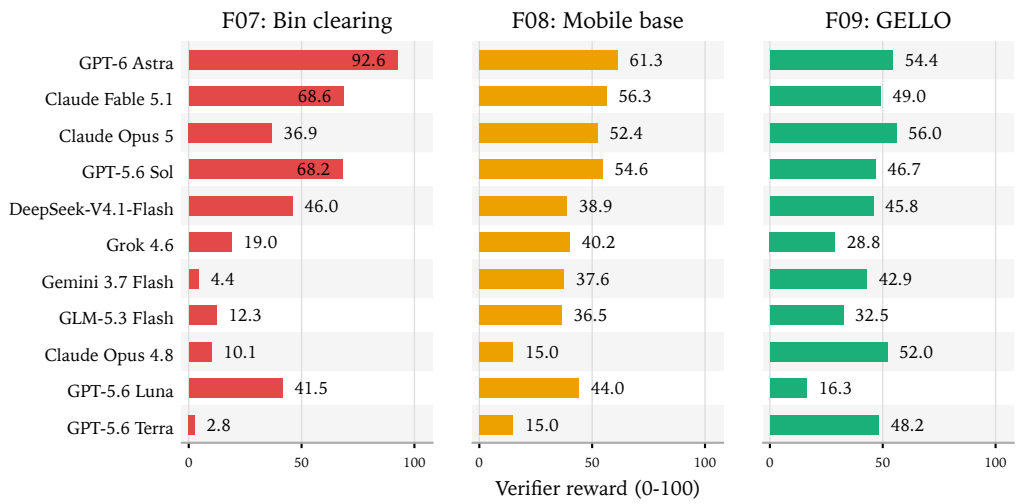


Figure 15: **Families 07–09: complete joint-task rewards.** Horizontal bars show bin clearing, mobile-base design, and GELLO gravity compensation. Each family contains one submitted joint task per system. Family 08 uses checkpoint-wise worst-arm credit and a gate cap; Family 09 averages the three arm rewards. Bars show the reported final reward with all gates applied.

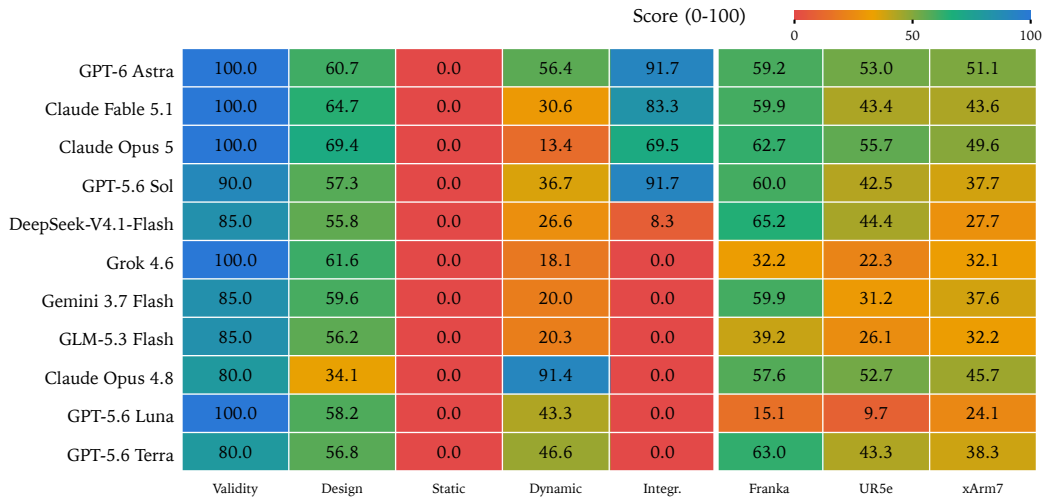


Figure 16: **Mechanical-design diagnostics.** The first five columns are Family 08 stage credits divided by their stage weights (0.15, 0.35, 0.15, 0.20, 0.15) and multiplied by 100. They precede the final gate cap and must not be substituted for the reward above. The last three columns are Family 09's individual arm rewards (0–100), whose arithmetic mean is the joint-task reward. Integr. denotes integration.

Design renderings. Figure 17 shows mobile-base designs, organized by model, alongside a reference solution.

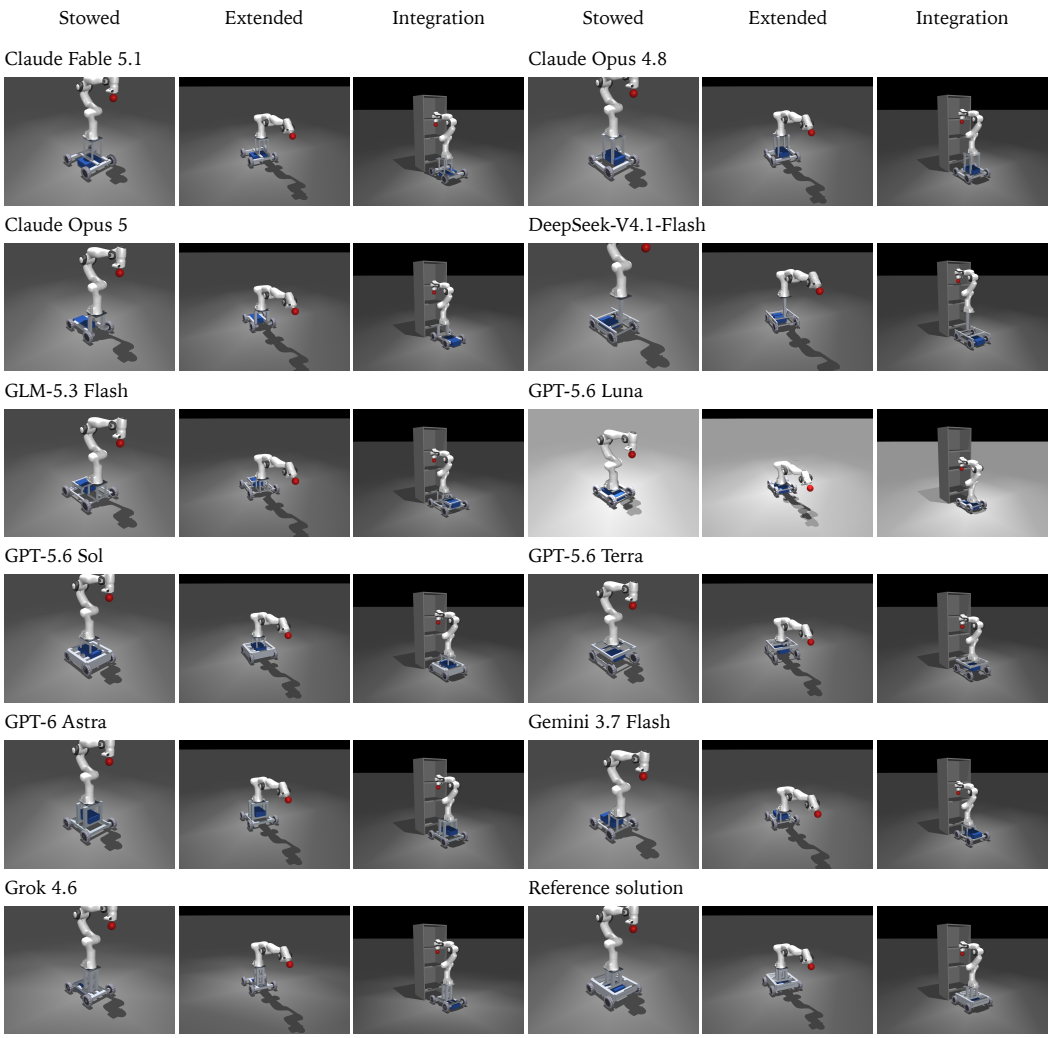


Figure 17: **Family 08: mobile-base design renderings.** The 36 panels show designs from all eleven evaluated models and a reference solution. Each labeled group contains stowed, extended, and integration views of one design. Verifier camera settings are preserved.

B.2 RESOURCE USE

Table 5: Resources and status for the retained runs. Costs are USD; elapsed time is in hours; token totals are in millions. Cost coverage counts runs with reported API costs. Grok 4.6 and Gemini 3.7 Flash token counts and total cost are absent due to the harness incompatibility with Harbor.

System	Mean $	Total $	Coverage	Mean h	Input M	Cached M	Output M	Failed
GPT-6 Astra	39.00	3899.09	51/51	2.42	2601.41	2552.76	12.07	0
Claude Fable 5.1	22.81	2246.63	51/51	2.50	2671.29	2630.17	22.51	0
Claude Opus 5	27.01	3143.92	51/51	2.53	5026.40	4968.04	27.91	0
GPT-5.6 Sol	17.17	1535.96	51/51	2.26	2872.33	2821.53	8.59	0
DeepSeek-V4.1-Flash	0.40	39.34	51/51	2.31	3372.05	3357.09	23.19	0
GLM-5.3 Flash	1.16	118.52	51/51	4.93	2994.11	2891.37	32.74	7
Claude Opus 4.8	22.79	2034.56	51/51	2.33	3105.33	3066.26	25.53	0
GPT-5.6 Luna	0.52	43.07	51/51	1.44	1525.54	1496.75	6.14	0
GPT-5.6 Terra	2.35	257.79	51/51	1.06	837.97	816.77	4.33	1

Cost and elapsed time. Mean API cost and elapsed time follow the same task–family–workflow hierarchy as the RLE Index. Total cost instead sums all retained runs. Elapsed time is the difference between run start and finish timestamps, not isolated agent compute. Failed-status counts describe run execution status, where all of them are timeout failures where the agent does not call termination or submission until the whole time budget is used up.